Selected Topics in Visual Recognition using Deep Learning Homework 2

Jia-Wei Liao, 309652008
Department of Applied Mathematics, NYCU
GitHub Link: https://github.com/Jia-Wei-Liao/SVHN_Dataset_Detection

March 9, 2022

Abstract

In this homework, we implement the deep learning method to detect the digits images. First, we use detection package to construct Faster R-CNN model [3]. After the hyper-parameter tuning, we obtain a testing mAP of 0.389141. It's not far from the baseline, but we can't beat it. So, in order to beat the baseline, we implement the YOLOv4 model [4]. We refer to the code on the github and write the data processing, submission code by ourselves. Finally, we exceed the baseline and get the test mAP of 0.41987.

1 Introduction

The Street View House Numbers (SVHN) Dataset is the most widely-used dataset for developing machine learning and object recognition. It's obtained from house numbers in Google Street View images. In this homework, we contains 33,402 training images and 13,068 test images in SVHN dataset and need to train a model to detect digits. The model should predict fast and precise on inference time.

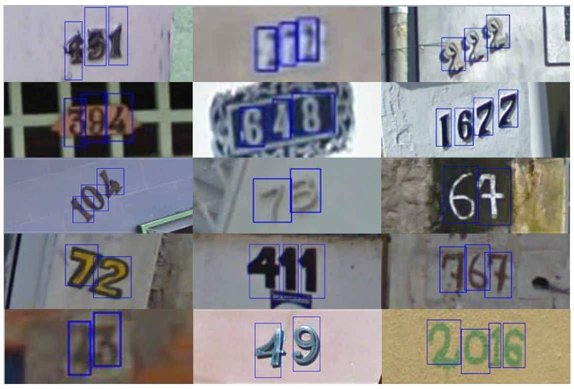


Figure 1. SVHN Dataset (from [6])

2 Data Pre-processing

2.1 Resize

Since the images in the dataset do not all have the same dimensions, we resize all images to the same size.

2.2 Data Augmentation

To avoid over-fitting, we do the data augmentation by flip with probability. Also, we use Mosaic data augmentation in YOLOv4. It combines many training images into one certain ratios. This allows for the model to learn how to identify objects at a smaller scale than normal. It is useful in training to significantly reduce the need for a large mini-batch size.

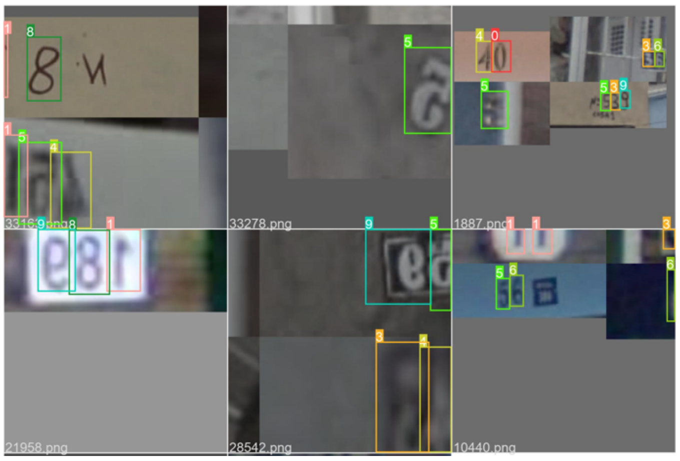


Figure 2. Mosaic data augmentation

3 Model architecture

3.1 Faster R-CNN

The Faster R-CNN improve the unnecessary feature extraction and apply ROI pooling to unify dimension. In order to accelerate detection, the author proposes RPN module which put the proposals into CNN.

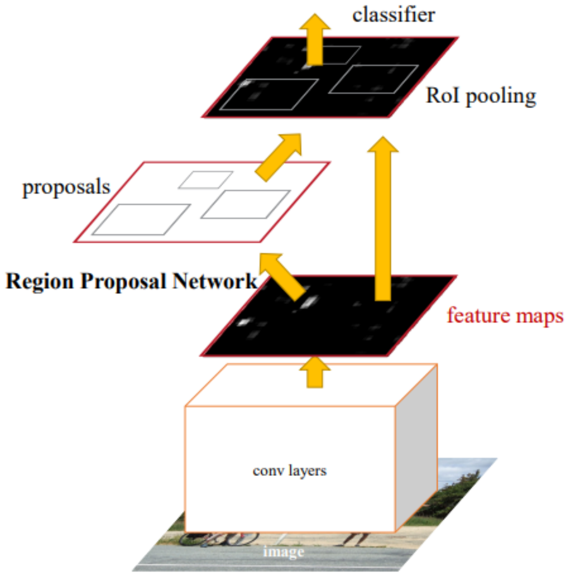


Figure 3. Faster R-CNN (from [3])

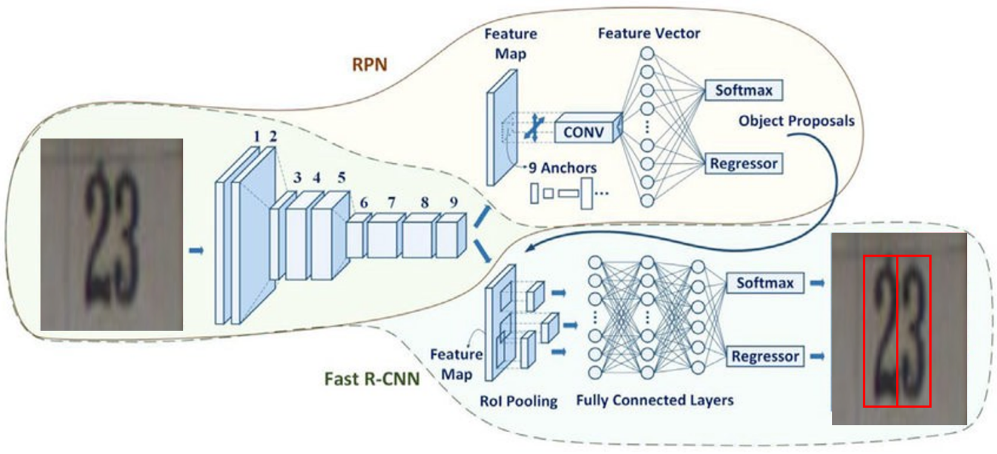


Figure 4. Faster R-CNN

3.2 YOLOv4

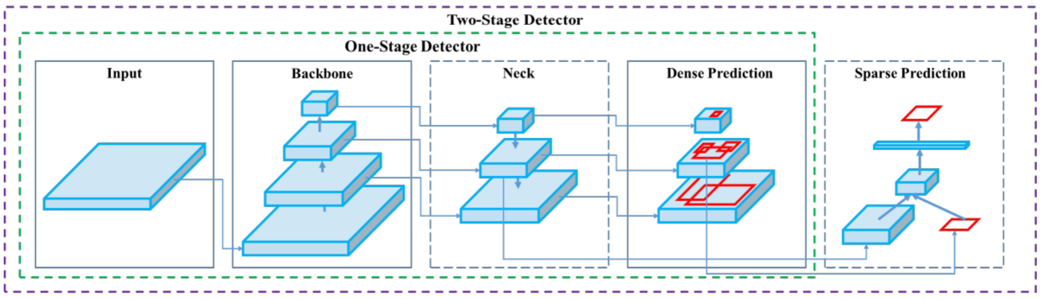


Figure 5. YOLOv4 Object detector (from [4])

- Backbone: Darknet53
- Neck: Feature Pyramid Networks (FPN)
- Head: YOLO

3.2.1 Darknet

The Darknet is CNN-based model which is based on residual block. It play the role of feature extraction in YOLOv4. Although it is similar to ResNet, Darknet is much faster than ResNet with about the same accuracy. So, it's suitable for use as a backbone.

3.2.2 Feature Pyramid Networks (FPN)

The FPN is a feature extractor. It can detect object at different scales since the smaller feature map has larger receptive field and the bigger feature map merged high resolution and low resolution with upsampling by addition. This can improve the accuracy of detecting small object.

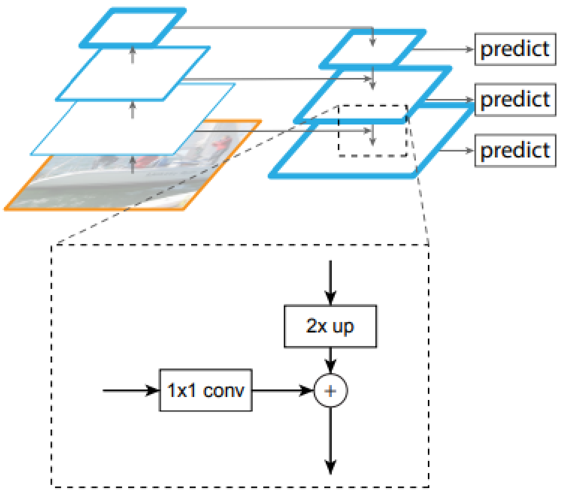


Figure 6. FPN block (from [1])

4 Hyperparameters Choice

- Batch size: 16
- Epoch: 50
- Optimizer: SGD
- Initial Learning Rate: 0.01
- Momentum: 0.937
- Weight Decay: 0.0005
- Learning Schedule: warm up

5 Losses for Object Detection

5.1 Box Loss

5.1.1 Intersection of Union (IoU)

Let B, B^{gt} be predicted box and ground truth box. The IoU Loss is defined as

$$\mathcal{L}_{\text{IoU}}(B, B^{gt}) = 1 - \frac{|B \cap B^{gt}|}{|B \cup B^{gt}|},$$

where $B = (x, y, w, h)$ is the predicted box and $B^{gt} = (x^{gt}, y^{gt}, w^{gt}, h^{gt})$ is is the ground-truth.

5.1.2 Complete-IoU (IoU)

The CIOU Loss is defined as

$$\mathcal{L}_{\text{CIoU}}(B, B^{gt}) = \mathcal{L}_{\text{IoU}}(B, B^{gt}) + \frac{d^2}{c^2} + \alpha v$$

where

- $d = \rho(b, b_{gt})$ is Euclidean distance between b and b_{gt}
- b and b^{gt} denote the central of B and B^{gt}
- c is the diagonal length of the smallest enclosing box covering the two boxes
- $\alpha = \dfrac{v}{\mathcal{L}_{\text{IoU}}(B, B^{gt}) + v}$
- $v = \dfrac{4}{\pi^2}\left[\tan^{-1}\left(\dfrac{w^{gt}}{h^{gt}}\right) - \tan^{-1}\left(\dfrac{w}{h}\right)\right]^2$

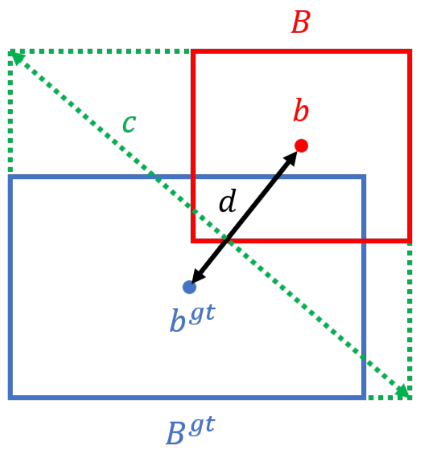


Figure 7. Bounding box for CIoU

The first term of CIoU is IoU loss, it illustrate the overlap rate of two boxes. The second part is describe the distance of the center point. And the third part is measure the similarity of width and height.

5.2 Classification Loss

5.2.1 Binary Cross Entropy Loss

Let y, y^{gt} be predicted category and ground truth category. For binary classification task, the cross entropy loss is defined as

$$\mathcal{L}_{\text{CE}}(y, y^{gt}) = -\left[y^{gt} \log y + (1 - y^{gt}) \log(1 - y)\right]$$

5.2.2 Focal Loss

The focal loss [2] is defined as

$$\mathcal{L}_{FL}(y, y^{gt}) = -(1 - p^t)^\gamma \log p_t$$

where $p_t = \begin{cases} y, & \text{if } y^{gt} = 1 \\ 1 - y, & \text{otherwise} \end{cases}$ and $\gamma \geq 0$ is called focusing parameter.

The candidate object or Anchor found in a picture has most of the proportions of the background rather than the foreground, so there will be a imbalance in calculating the loss. The focal loss is down-weighted for the easy example. The hard example can be trained as much as possible during the training process and ignored those easy examples. So it can solve the category imbalanced problem.

5.3 Loss for YOLOv4

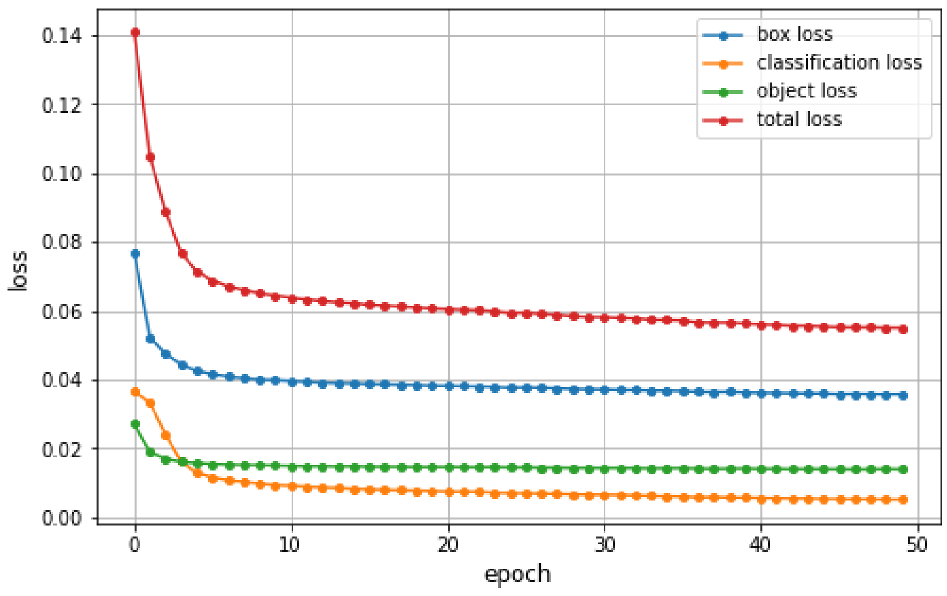


Figure 8. Loss for YOLOv4 on 50 epoch

6 Result

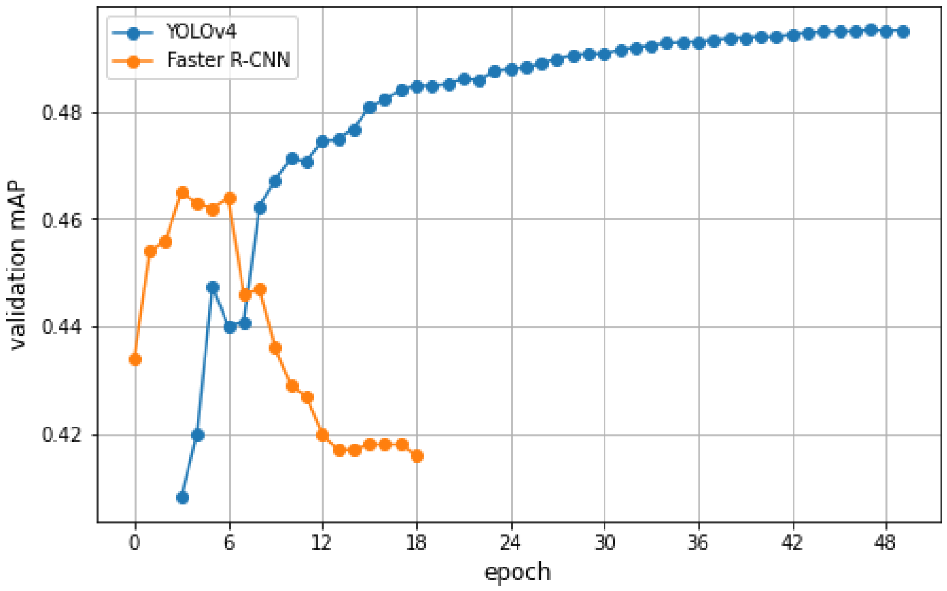


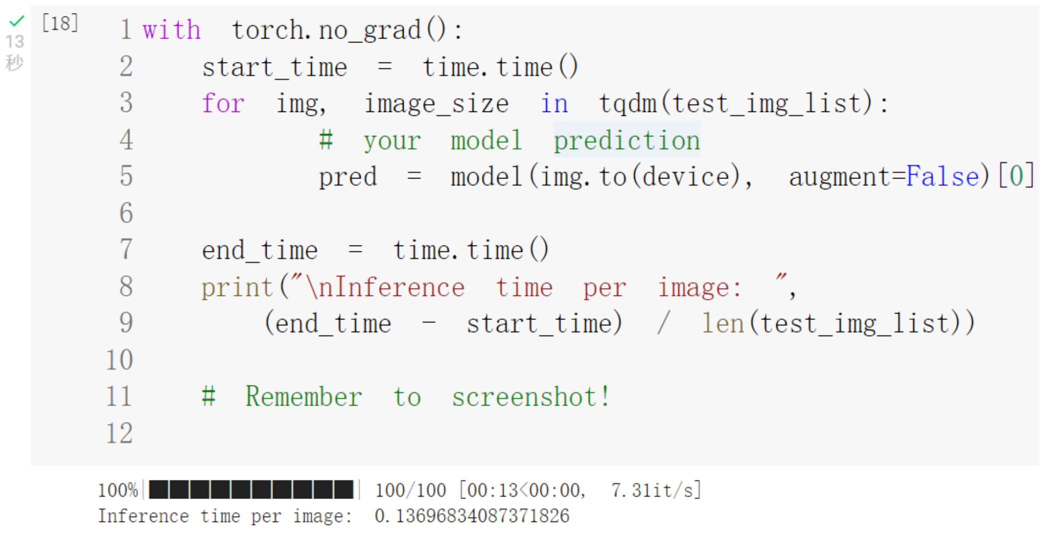
Figure 9. mAP$_{50:95}$ in validation step

Model	Faster RCNN	YOLOv4
training time for per epoch	57 min	27 min
(epoch, best mAP$_{50:95}$)	(4, 0.465)	(50, 0.495)
test mAP$_{50:95}$	0.38914	0.41987
time on P100 GPU (image /per s)	0.07364	0.2
time on Colab K80 GPU (image /per s)	0.13696	×

Table 1. mAP$_{50:95}$

7 Colab

We use Tesla K80 GPU on Colab to inference the model predicted time per second and provide link: https://colab.research.google.com/drive/1iosQjMUfzmDVLkrXhI13IZyuIJuTiArq?usp=sharing.



```
✓   [18]
13
秒
with torch.no_grad():
    start_time = time.time()
    for img, image_size in tqdm(test_img_list):
        # your model prediction
        pred = model(img.to(device), augment=False)[0]

    end_time = time.time()
    print("\nInference time per image: ",
        (end_time - start_time) / len(test_img_list))

    # Remember to screenshot!

100%|████████████| 100/100 [00:13<00:00,  7.31it/s]
Inference time per image:  0.13696834087371826
```

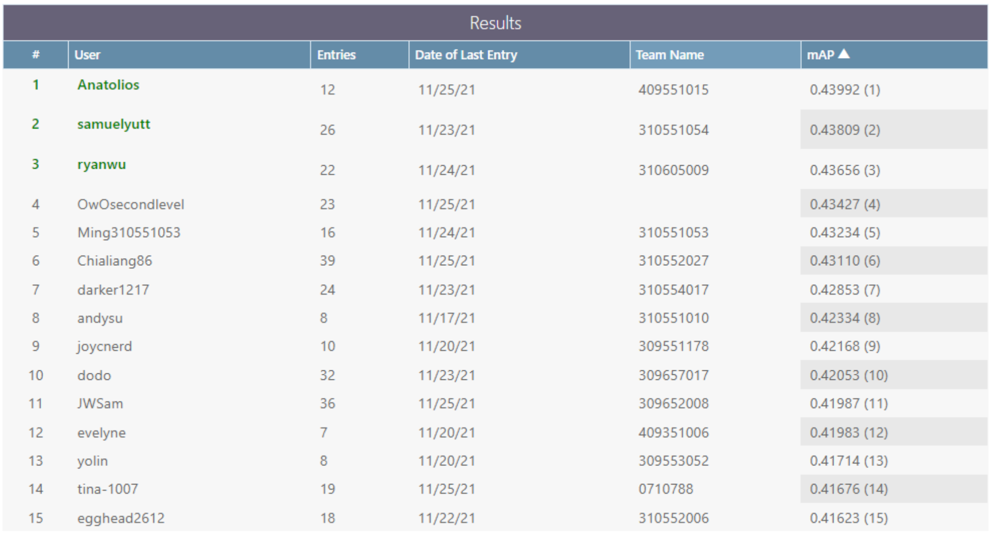
Figure 10. Inference time on Colab

8 Summary

In this homework, we implement Faster R-CNN and YOLOv4 model. First, we train the Faster R-CNN for 20 epochs. and we try the different optimizers, SGD, Adam and AdamW. The results show that the loss of AdamW can drop quickly and steadily. Hence we choose the AdamW as our strategy. After hyperparameter turning, the model get the best mAP for 4 epochs. Afterwards it start to overfitting (the loss keep falling but the mAP of validation begin descending). We get the best validation mAP of 0.465 but obtain test mAP of 0.389141 after upload to the system.

To beat the baseline, we implement the YOLOv4 model from github. The author of YOLOv4 is from Academia Sinica in Taiwan, so we are curious about it. We study the paper and modify the code. As the same as our previous experiment, we try the different optimizers. but Adam got a worst performance in this time and SGD has a good performance. So we choose the SGD and train the 50 epochs. Aftering 25 hours, we the best validation mAP of 0.495 and test mAP of 0.41987.

In terms of speed, YOLOv4 predict much faster than Faster R-CNN. On our device (P100 GPU), their speed difference is about 2.7 times. No matter in terms of speed or accuracy, YOLOv4 wins Faster R-CNN.



Results

#	User	Entries	Date of Last Entry	Team Name	mAP ▲
1	Anatolios	12	11/25/21	409551015	0.43992 (1)
2	samuelyutt	26	11/23/21	310551054	0.43809 (2)
3	ryanwu	22	11/24/21	310605009	0.43656 (3)
4	OwOsecondlevel	23	11/25/21		0.43427 (4)
5	Ming310551053	16	11/24/21	310551053	0.43234 (5)
6	Chialiang86	39	11/25/21	310552027	0.43110 (6)
7	darker1217	24	11/23/21	310554017	0.42853 (7)
8	andysu	8	11/17/21	310551010	0.42334 (8)
9	joycnerd	10	11/20/21	309551178	0.42168 (9)
10	dodo	32	11/23/21	309657017	0.42053 (10)
11	JWSam	36	11/25/21	309652008	0.41987 (11)
12	evelyne	7	11/20/21	409351006	0.41983 (12)
13	yolin	8	11/20/21	309553052	0.41714 (13)
14	tina-1007	19	11/25/21	0710788	0.41676 (14)
15	egghead2612	18	11/22/21	310552006	0.41623 (15)

Figure 11. Leaderboard on CodaLab

References

[1] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie, Feature Pyramid Networks for Object Detection, *CVPR*, 2017.

[2] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár, Focal Loss for Dense Object Detection, *CVPR*, 2017.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks *CVPR*, 2016.

[4] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection *CVPR*, 2020.

[5] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, Dongwei Ren Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression *CVPR*, 2019.

[6] SVHN Website: `http://ufldl.stanford.edu/housenumbers/`